

Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method of loading a trustable operating system comprising:
performing a start secure operation by a first processor of a plurality of processors;
performing a join secure operation by remaining processors of the plurality of processors
excluding the first processor, the join secure operation performed atomically from the start
secure operation and forces the remaining processors of the plurality of processors to enter into a
halted state that prevents the remaining processors from interfering with the operations of the
first processor;
receiving signals by the first processor from the remaining processors that the remaining
processors have entered the halted state;
identifying a secure region in a memory of a computer;
loading a content into the identified region under control by the first processor after
receiving the signals that the remaining processors have entered the halted state;
registering an identity of the content after the content is loaded into the identified region,
the registering comprises:
recording a hash digest of the content of the identified region, and
signing the hash digest with a hash signing engine having a secure channel to
access the hash digest, the signed hash digest being stored in a register in the memory of
the computer that is accessible by an outside entity to verify whether the content can be
trusted;
causing the first processor to jump to a known entry point in the identified region in the
memory; and
completing the start secure operation by the first processor and signaling the remaining
processors to resume activity by exiting the halted state and jumping to the known entry point in
the identified region in the memory.

2. (Previously Presented) The method of claim 1, further comprising:

preventing interference with the identifying, loading, and registering by at least a second processor of the plurality of processors while the first processor is loading the content into the identified region.

3. (Previously Presented) The method of claim 2, wherein preventing interference comprises halting at least the second processor of the plurality of processors until the identifying, loading, and registering is complete.

4. (Currently Amended) The method of claim 1, further comprising:
~~blocking access to the secure region of the memory for a duration of the start secure operation even after receiving the signals that the remaining processors have entered the halted state when the plurality of processors are implemented within a computer system that supports direct memory access (DMA) causing at least the second processor of the plurality of processors to jump to the known entry point in the identified region.~~

5. (Previously Presented) The method of claim 1, wherein identifying comprises receiving a region parameter, the region parameter specifying a location of the region.

6. (Previously Presented) The method of claim 5, wherein the location comprises a range of addresses in the memory of the computer within which the region is located.

7. (original) The method of claim 5, wherein the location comprises a start address and a length of the memory of the computer within which the region is located.

8. (canceled)

9. (original) The method of claim 1 wherein the content is a component of an operating system to operate the computer.

10. (Previously Presented) The method of claim 9, wherein the component of the operating system is a one of a virtual machine monitor, and a privileged software nucleus.

11. (original) The method of claim 1 wherein identifying, loading and registering are uninterruptible.

12. (Currently Amended) An article of manufacture comprising:
a machine-accessible medium including a data that, when accessed by a machine cause the machine to,

halt all but one of a plurality of central processing units (CPU_s) in a computer;

identify a region in a memory of the computer;

block access to the identified region by all resources except the non-halted CPU only after receiving signals by the one of the plurality of CPUs that a remainder of the plurality of CPUs have entered into a halted state;

load a content into the identified region;

register an identity of the content of the identified region, the registering comprises:

computing the cryptographic hash of the identified region, recording the computed cryptographic hash of the content in the identified region, and

signing the computed cryptographic hash with a hash signing engine having a secure channel to access the cryptographic hash, the signed cryptographic hash being stored in a register in the memory of the computer that is accessible by an outside entity to verify whether the content can be trusted; and

cause the non-halted CPU to begin executing at a known entry point in the identified region after the identity of the content has been registered.

13. (original) The article of manufacture of claim 12, wherein the data that causes the machine to halt the all but one of a plurality of CPUs comprises data causing the all but one of a plurality of CPUs to enter a halted state.

14. (Previously Presented) The article of manufacture of claim 13, wherein the data further causes the halted CPUs to exit the halted state after the one of the plurality of CPUs has begun executing at the known entry point in the identified region

15. (original) The article of manufacture of claim 14, wherein the data further causes the previously halted CPUs to begin executing at the known entry point in the identified region upon exiting the halted state.

16. (Previously Presented) The article of manufacture of claim 13, wherein the data that causes the machine to record the cryptographic hash includes data that further causes the machine to,

erase a hash digest area in the memory of the computer; and
record a platform information in the hash digest area; the platform information includes a version number of the one of the plurality of CPUs.

17. (canceled)

18. (original) The article of manufacture of claim 13, wherein the data that causes the machine to identify the region in memory of the computer includes data that further causes the machine to receive at least one region parameter containing a location of the identified region.

19. (original) The article of manufacture of claim 13, wherein the location includes an address of the identified region.

20. (original) The article of manufacture of claim 13, wherein the location includes a length of the identified region.

21. (Currently Amended) A method of securing a region in a memory of a computer comprising:

halting all but one of a plurality of processors in a computer, the plurality of ~~processors~~ ~~halted processors~~ entering into a special halted state;

identifying a region in a memory of a computer;

loading content into the region ~~only after the halting of all but the one of the plurality of processors~~;

blocking access to the region in a memory of the computer by all resources except the non-halted processor;

registering an identity of the content of the region in the memory, the registering comprises:

recording a cryptographic hash of the region, and;

signing the cryptographic hash with a digest signing engine coupled to the memory of the computer having a secure channel to access the cryptographic hash, the signed cryptographic hash being stored in a register in the memory of the computer that is accessible by an outside entity to verify whether the content can be trusted; and placing the non-halted processor into a known privileged state; releasing the halted processors after the non-halted processor has been placed into the known privileged state.

22. (Previously Presented) The method of claim 21, further comprising causing the non-halted processor to jump to a known entry point in the region.

23. (Canceled).

24. (Previously Presented) The method of claim 22, further comprising causing the halted processors to exit the special halted state so as to release the halted processors after the non-halted processor has been placed into the known privileged state.

25. (Previously Presented) The method of claim 21, further comprising causing the previously halted processors to begin executing at a known entry point in the region upon exiting the special halted state.

26. (Previously Presented) The method of claim 21, wherein recording the cryptographic hash comprises:

erasing a hash digest area in the memory of the computer;
recording a platform information in the hash digest area, the platform information including a version number of the non-halted processor;
computing the cryptographic hash of the content of the region; and
recording the computed cryptographic hash in the hash digest area.

27. (original) The method claim 26, wherein the hash digest area is a register in the memory of the computer.

28. (canceled)

29. (original) The method of claim 21, wherein the region is specified in at least one region parameter.

30. (original) The method of claim 29, wherein the at least one region parameter is an address of the region in the memory of the computer that is to be secured.

31. (original) The method of claim 29, wherein the at least one region parameter is a length of the region in the memory of the computer that is to be secured.

32-38. (Canceled).

39. (Currently Amended) A method of loading a trustable operating system comprising:
selecting an area in a memory accessible to a first processor of a plurality of processors,
the plurality of processors including the first processor and at least one processor;

halting all processors of the plurality of processors except for the first processor from
accessing the memory;

loading data into the selected area after the first processor receiving signaling from the at
least one processor to indicate that the at least one processor is in a halted state;

registering an identity of the data loaded in the selected area by

recording a unique cryptographic function of the data loaded in the selected area,
and

signing the unique cryptographic function with a hash signing engine having a
secure channel to access the unique cryptographic function, the signed unique
cryptographic function being stored in a register in memory and accessible by an outside
entity to verify whether the data is trustworthy;

directing the first processor to commence processing at an entry point in the selected
area; and

releasing all of the halted processors and directing the released processors to commence
processing at the entry point of the selected area.

40. (Previously Presented) The method of claim 39, wherein preventing interruption comprises halting any other processors having access to the memory until the selecting, loading, and directing is complete.

41. (Previously Presented) The method of claim 40, further comprising:
causing the other processors to commence processing at an entry point in the selected area.

42. (Previously Presented) The method of claim 39, wherein selecting comprises receiving a parameter specifying a location of the selected area.

43. (Previously Presented) The method of claim 42, wherein the location is a range of addresses in memory within which the selected area is located.

44. (Previously Presented) The method of claim 42, wherein the location comprises a start address and a length of memory within which the area is located.

45. (cancelled)

46. (Previously Presented) The method of claim 39 wherein the data is a component of an operating system to operate a device in which the memory resides.

47. (Previously Presented) The method of claim 46, wherein the operating system has a graphical user interface.